基于类的动态依赖关系的集成测试顺序分配策略

李小将[1], 李佑禄[2], 陈启安[3]
(1. 装备指挥技术学院 试验工程系, 北京 101416; 2. 空军94981部队, 江西 南昌 330100;
3. 厦门大学 计算机系, 福建 厦门 361052)

摘 要: 类的集成测试顺序决定着测试工作量的大小。通过分析面向对象程序中类之间的完整依赖关系, 给出了基于测试级的集成测试顺序分配策略及其实现。首先, 分析了面向对象程序的类之间的依赖关系, 并给出了数学描述; 接着给出了测试级的定义以及基于测试级的测试顺序分配策略和算法; 最后, 介绍了基于该分配策略的一个测试级顺序自动生成工具——TLOGOS。该工具已被用于YSS2000电力监控组态系统的测试, 大大减少了测试的工作量。

关 键 词: 测试级; 类; 集成测试; 顺序分配
中图分类号: TP 311.52
文献标识码: A 文章编号: 1673-0127(2005)01-0093-05

An Order-assigned Strategy of Integration Testing Based on the Dynamic Dependency Relation of Classes

LI Xiao-jiang[1], LI You-lu[2], CHEN Qi-an[3]
(1. Department of Test Engineering, the Academy of Equipment Command & Technology, Beijing 10141, China;
2. No. 94981 of Air Force, Nanchang Jiangxi 330100, China;
3. Department of Computer Science, Xiamen University, Xiamen Fujian 361052, China)

Abstract: The order of class integration testing decides the amount of testing work. In the paper, an order-assigned strategy of class integration testing is presented and implemented by analyzing the complete dependency relation of classes in OO program. Firstly, it analyzes the dependency relation of classes in OO program and gives the math description. Secondly, it defines the testing level of all classes, and proposes order-assigned strategy of class integration testing based on it. At last, it introduces a testing level order automatic generating tool — TLOGOS, which is implemented on the basis of the above strategy. The tool has been used in YSS2000 Electric Power Inspect and Control System, and reduced the testing work obviously.

Key words: testing level; classes; integration testing; order-assigned

选择不同的测试顺序将决定着测试工作量的大小, 如何寻找使得测试工作量最小的测试顺序是面向对象软件集成测试的一个重要问题[1]。Kung 等人在文献[2]中提出了一个对象关系模型, 不仅考虑了类之间的继承关系, 而且考虑了类之间的聚集关系和关联关系, 给出了一种集成测试顺序分配方法。但该测试顺序分配方法存在一个重要缺陷: 只考虑了类之间的静态依赖关系, 而

收稿日期: 2004-01-15
基金项目: 国家"863"计划基金资助项目(2001AA115090)
作者简介: 李小将(1973-), 男(汉族), 江西南昌人, 讲师, 博士, Ftq-Lxj@163.com.

没有考虑类之间的动态依赖关系。本文针对 Kung 等人给出的类的集成测试顺序分配方法存在的缺陷,通过分析面向对象程序中类之间的完整依赖关系,提出了一种基于测试级类的集成测试顺序分配策略,并实现了该策略。

1 类之间的依赖关系及数学描述

首先在 Kung 等人提出的对象关系图基础上,给出类之间静态依赖关系的三元组表示;然后,在静态依赖关系的基础上,进一步分析了类之间的动态依赖关系以及数学描述。

1.1 类之间的静态依赖关系及三元组表示

Kung 等人的测试顺序分配方法是基于他们提出的对象关系图(object relation graph, ORG)。一个面向对象程序 P 的 ORG 是一个带标签边的有向图,如图 1 所示。

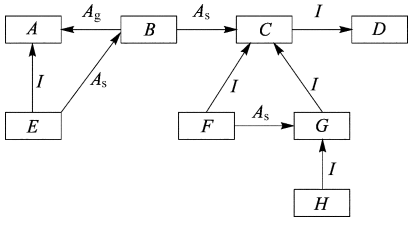

图 1 面向对象程序 P 的 ORG

其中,图中节点表示 P 中的类,带标签的边表示 P 中类之间的继承、聚集和关联关系。对于 P 中的任何 2 个类 C_1 和 C_2:

1) 一条从 C_1 到 C_2 标有 I 的边表示 C_1 是 C_2 的子类;

2) 一条从 C_1 到 C_2 标有 A_g 的边表示 C_1 是 C_2 的一个聚集类(C_1 包含一个或多个 C_2 对象);

3) 一条从 C_1 到 C_2 标有 A_s 的边表示 C_1 与 C_2 关联。

因而,ORG 描述的是 P 中的类之间的静态(编译阶段)依赖关系。例如,一个 OO 程序 P 包含 8 个类$\{A, \cdots, H\}$,图 1 是该程序的 ORG。其中,A 是 E 的一个父类,B 是 A 的聚集类,并且 B 关联 C 等等。

对于 ORG 图中的每个类和其他类的关系,可用一个三元组 R 表示

$$R = (C, I, O)$$

式中:C 为 ORG 中的某个类;I 为一个三元组,且

$$I = (I_i, I_{ag}, I_{as})$$

其中:I_i 为 ORG 中类 C 的带标签 I 的进边所连的类的集合;I_{ag} 为 ORG 中类 C 的带标签 A_g 的进边所连的类的集合;I_{as} 为 ORG 中类 C 的带标签 A_s 的进边所连的类的集合。O 为一个三元组,且

$$O = (O_i, O_{ag}, O_{as})$$

其中:O_i 为 ORG 中类 C 的带标签 I 的出边所连的类的集合;O_{ag} 为 ORG 中类 C 的带标签 A_g 的出边所连的类的集合;O_{as} 为 ORG 中类 C 的带标签 A_s 的出边所连的类的集合。按照上述表示,图 1 中的类之间的关系描述见表 1。

表 1 ORG 图中类的关系三元组

类	关系三元组
A	$(A, (\{E\}, \{B\}, \cup), (\cup, \cup, \cup))$
B	$(B, (\cup, \cup, \{E\}), (\cup, \{A\}, \{C\}))$
C	$(C, (\{F, G\}, \cup, \{B\}), (\{D\}, \cup, \cup))$
D	$(D, (\{C\}, \cup, \cup), (\cup, \cup, \cup))$
E	$(E, (\{\cup, \cup, \cup\}, (\{A\}, \cup, \{B\}))$
F	$(F, (\{\cup, \cup, \cup\}, (\{C\}, \cup, \{G\}))$
G	$(G, (\{\{H\}, \cup, \{F\}\}, (\{C\}, \cup, \cup))$
H	$(H, (\cup, \cup, \cup), (\{G\}, \cup, \cup))$

1.2 类之间的完整依赖关系及数学描述

上述 ORG 图和三元组表示只是考虑了类之间的静态依赖关系,而没有考虑类之间的动态依赖关系。分析类之间的包含静态和动态依赖的完整依赖关系,用集合和函数来描述类之间的静态和动态关系,并在 ORG 的基础上,给出类之间完整关系图 ODRG(object dynamic relation group)。

对于面向对象集成测试中的每一个类,可用 2 个类的集合和一个函数对类的测试依赖关系进行描述。

1) $D_1(X)$:表示 X 在编译阶段静态依赖的类的集合。

2) $D_2(X)$:表示 X 在编译阶段静态依赖和在执行阶段动态依赖的类的集合。

3) $B_d(X)$:是一个布尔函数,表示 X 是否至少动态依赖 $D_2(X)$ 中的一个类,如果是,$B_d(X)$ 的值为 1,否则,$B_d(X)$ 的值为 0。

下面,假定所给出的 ORG 图为已断开循环的无环有向图,得出 $D_1(X)$, $D_2(X)$ 和 $B_d(X)$ 的形式化定义。

定义 **1** 对于一个 ORG 中的类 X 和 C_k, X 静态依赖 C_k,当且仅当在 ORG 中存在一条从 X 到 C_k 的有向路径。令 R_s 表示 ORG 中一条有向边的二元关系:$R_s = \{(C_1, C_2) \mid$ 在 ORG 中存在一条从 C_1 到 C_2 边$\}$。则

$$D_1(X) = \{C_k \mid < X, C_k > \in R_s^+\}$$

其中:R_s^+ 表示 R_s 的传递闭包。

类之间的动态依赖关系可从静态依赖关系得到,例如:在图 1 中,B 关联 C, C 是 F, G, H 的父

类, 在执行阶段, B 可能关联 F, G, H。

一般地, 若 C_j 是 X 的一个服务类, X 关联 C_j 或是 C_j 的一个聚集类, 在执行阶段, X 可能依赖 C_j 所有的子类, 包括直接的子类或子类的子类。

定义 **2**　类之间的动态依赖关系用 R_d 表示, 令

$$R_d = \{< C_1, C_3 > \mid$$

存在一个类 C_2, 它既是 C_1 的服务类, 又是 C_3 的父类$\}$, 则

$$D_2(X) = \{ C_k \mid < X, C_k > \in (R_s \cup R_d)^+ \}$$

在 ORG 中增加表示动态关系的边, 用标有 D 的边表示, 可得到一个包含静态和动态依赖关系的完全的 ORG, 称为 ODRG。图 1 ORG 的 ODRG 如图 2 所示, 尽管原来的 ORG 中静态依赖已断开有向环路, 然而, 由于动态依赖的引入, 使得 ODRG 图中构成了新的环路, 也就是说, 可能有 $X \in D_2(X)$。在这种情形下, 构成循环中的所有类都具有相同的 $D_2(X)$, 如图 2 中由于引入了一条从 B 到 E 的动态依赖边, 使得 B 和 E 构成一个环路:

$$D_2(B) = D_2(E) = \{A, B, C, D, E, F, G, H\}$$

图 2 中的静态依赖集合, 动态依赖集合及布尔函数如表 2 所示。

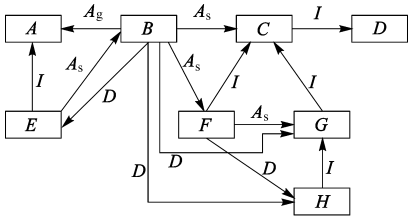

图 2　面向对象程序 P 的 ODRG

表 **2**　图 **2** 中 **ODRG** 的 $D_1(X)$、$D_2(X)$ 和 $B_d(X)$

类 X	$D_1(X)$	$D_2(X)$	$B_d(X)$
A	⌀	⌀	0
B	$\{A, C, D\}$	$\{A, B, C, D, E, F, G, H\}$	1
C	$\{D\}$	$\{D\}$	0
D	⌀	⌀	0
E	$\{A, B, C, D\}$	$\{A, B, C, D, E, F, G, H\}$	1
F	$\{C, D, G\}$	$\{C, D, G, H\}$	1
G	$\{C, D\}$	$\{C, D\}$	0
H	$\{C, D, G\}$	$\{C, D, G\}$	0

定义 **3**　对于 ODRG 图中的所有的类 X, $D_1(X)$ 是 $D_2(X)$ 的子集, $B_d(X)$ 是一个关于 X 的布尔函数, 如果 $D_1(X)$ 是 $D_2(X)$ 的真子集, 则 $B_d(X)$ 的值为 1; 否则, $B_d(X)$ 的值为 0。

2　基于测试级的测试顺序分配策略

本节在对类之间完整关系分析的基础上, 给出测试级的定义策略和基于测试级的类的集成测试顺序分配策略及实现算法。

2.1　测试级的定义

为定义 ODRG 中的测试级, 采用以下策略。对测试的目标类按照静态依赖和动态依赖分为 2 个级别: 对于每个类, 将各个类的静态依赖测试定义为一个测试级; 而动态依赖测试定义为另一个测试级。一个测试级 T 用一个三元组表示为

$$T = (T.\text{goal}, T.\text{need}, T.\text{type})$$

式中: $T.\text{goal}$ 为需测试目标类的集合; $T.\text{need}$ 为测试目标类时所涉及的所有类的集合, 包括目标类和根据测试类型所涉及的所有类; $T.\text{type}$ 为测试的依赖类型。静态依赖测试用 Sta 表示, 动态依赖用 Dyn 表示。

一个 ODRG 中的类的各个测试级可从 $D_1(X)$、$D_2(X)$ 和 $B_d(X)$ 直接得到: $D_1(X)$ 给出了测试类 X 所需要类的最小集合, 仅仅考虑 X 静态依赖的类, 而不考虑动态依赖的类; $D_2(X)$ 给出了测试类 X 所需要类的最大集合; $B_d(X)$ 表示测试级的类型为 Sta, 还是 Dyn。如果

$$B_d(X) = 1$$

则测试级类型为 Dyn; 否则为 Sta。所有的测试级按照以下方式进行定义。

1) 对于 ODRG 中的每一个类, 分别为每个类定义一个静态测试级

$$T = (\{X\}, \{X\} \cup D_1(X), \text{Sta});$$

2) 对于 ODRG 中 $B_d(X)$ 的值为 1 的测试类定义一个动态测试级

$$T = (T.\text{goal}, T.\text{need}, \text{Dyn})。$$

当测试一个类 X 时,

$$T.\text{need} = \{X \cup D_2(X)\}。$$

考虑到在 ODRG 中可能存在环路, 即可能存在 $X \in D_2(X)$, 则在环路中的所有类都具有相同的 $T.\text{need}$, 只需要为环中的所有类定义一个测试级, 而将所有的类加入到 $T.\text{goal}$ 中。因而, 动态测试级定义分为以下 2 种情况:

① 如果不存在包含 X 的环路, 则

$$T = (\{X\}, X \cup D_2(X), \text{Dyn});$$

② 如果存在包含 X 的环路, 则

$$T = (\{X_1, \cdots, X_j\}, D_2(X_1), \text{Dyn})。$$

其中, $X_1, \cdots, X_j$ 为处于相同环中的所有类。

按上述方式, 可定义图 2 中 ODRG 的所有测试级如表 3 所示。前 8 行表示静态测试级, 后 2 行表示动态测试级, 因 B 和 E 构成一个环路, 所以将 B 和 E 的动态测试级合并形成一个测试级。

表 3 图 2 中 ODRD 的测试级

T = (T.goal, T.need, T.type)		
T.goal	T.need	T.type
{A}	{A}	Sta
{B}	{A, B, C, D}	Sta
{C}	{C, D}	Sta
{D}	{D}	Sta
{E}	{A, B, C, D, E}	Sta
{F}	{C, D, F, G}	Sta
{G}	{C, D, G}	Sta
{H}	{C, D, G, H}	Sta
{B, E}	{A, B, C, D, E, F, G, H}	Dyn
{F}	{C, D, F, G, H}	Dyn

2.2 测试级测试顺序分配策略

分配测试级测试顺序的目标是使得在测试一个类之前, 该类所依赖的所有类都已经测试, 而且在对一个类进行动态测试之前, 所有的静态测试都已经进行。根据这一目标, 我们给出测试级的顺序分配规则。

规则 M = (M.goal, M.need, M.type) 和 T = (T.goal, T.need, T.type) 分别为 2 个测试级, 若满足以下 2 个条件之一:

1) M.type= T.type并且M.need是T.need的真子集;

2) M.type= Sta, 且 T.type= Dyn, M.need是 T.need 的子集。

则称测试级 M 在测试级 T 之前。

上述规则包含以下 4 种情况:

1) 如果类 X 静态依赖 Y, 则对 Y 的静态测试级先于对 X 的静态测试级。

2) 如果类 X 有 2 个测试级。静态测试级 T_s = ({X}, $X \cup D_1(X)$, Sta) 和动态测试级 T_d = (goal, $X \cup D_2(X)$, Dyn), 其中, $X \in$ goal。则 T_s 先于 T_d。

3) T_x = (T_x.goal, T_x.need, Dyn) 是类 X 的一个动态测试级, 并且 $Y \in T_x$.need, $X \neq Y$。则对 Y 的静态测试级 T_y = ({Y}, $Y \cup D_1(Y)$, Sta) 先于 T_x。

4) 类 X 和类 Y 有2个不同的动态测试级 T_x 和 T_y, 如果 X 动态依赖 Y, 则 T_y 先于 T_x。

根据该规则, 下面采用类 C 程序风格给出一个 ODRG 图的测试级测试顺序分配算法的简单描述。

```
输入: 一个 ODRG 的所有测试级;
输出: 经过排序的测试级;
  int  n; //表示测试级的个数;
  Test_Level  T[n]; //Test_Level 表示测试级结构类型,
  T[n]存放测试级队列;
  int  pre: 存放临时较先测试级的在队列中的位置
  for(i= 1; i< = n, i+ + )
  {  T[pre] = T[0];
    for(j= 1, j < = i; j + + )
      {
        按照规则 1 比较 T[ pre] 和 T[j];
        if(T[j] 先于 T[pre])
        then pre= j;
      }
    if(pre! = i)
    then 交换 T[pre] 和 T[j];
}
```

用一个有向图 G = [V, E] 表示一个 ODRG 图的测试级顺序, 其中, 节点代表测试级; 边表示测试优先关系。采用上述算法获得表 3 中测试级的顺序如图 3 所示。

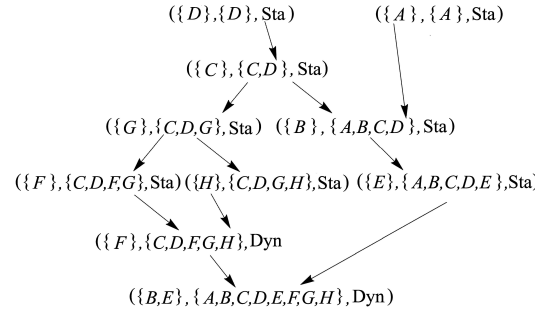

图 3 表 3 中测试级测的试顺序分配图

3 测试顺序生成工具 TLOGOS[3~5]

TLOGOS(test level order generator for object-orient system) 是上述类测试级分配测试的一个实现工具, 其功能结构如图 4 所示。

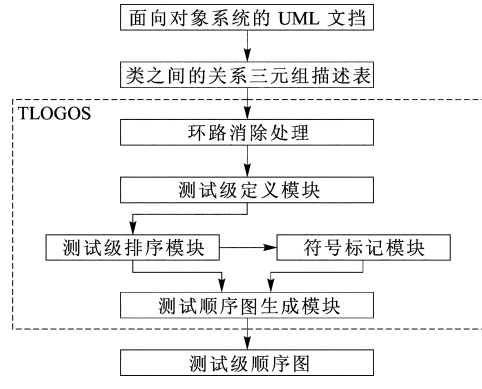

图 4 TLOGOS 功能结构图

该工具的输入信息是一个描述面向对象系统中类的关系三元组列表。该列表可以手工输入, 也可以根据面向对象系统的统一建模语言(uniform model language, UML) 设计文档中的 UML 类图获得。下面介绍 TLOGOS 的几个主

要功能模块。

1) 环路消除模块: 根据输入的关系三元组描述信息所表示的 ORG, 采用深度优先搜算法, 依次找出该 ORG 图中的非单元类簇, 通过断开一条或多条关联边, 将非单元类簇的环路断开, 直到 ORG 中所有的环路消除为止, 以得到新的无环的 ORG 的关系三元组描述。

2) 定义测试级模块: 根据新的关系三元组, 按照上述类之间的测试依赖关系分析方法, 对每个类生成相应的 $D_1(X)$、$D_2(X)$ 和 $B_d(X)$; 然后根据每个类的 $D_1(X)$、$D_2(X)$ 和 $B_d(X)$, 按照 2.1 中介绍测试级的定义方法, 定义所有的测试级。

3) 测试级排序模块: 对定义测试级模块获得的所有的测试级, 按照 2.2 节中测试级的顺序分配策略, 对所有的测试级进行排序, 以生成测试级的顺序。

4) 测试级顺序图生成模块: 将测试级顺序用测试级顺序图的方式显示出来。

TLOGOS 能够自动生成一个面向对象系统中的测试级分配顺序, 根据该测试级分配顺序进行测试, 可减少测试的工作量。然而, 通过手工的方式, 从 UML 文档获取类之间的关系三元组信息是一项繁杂的工作, 且容易出错。目前, 笔者正在进一步研究 UML 类图到类的关系三元组的自动生成方法。这样, TLOGOS 将会更加实用。

4 结　论

由于面向对象软件程序中, 类之间依赖关系的复杂性, 使得面向对象软件的集成测试非常困难。合理的集成测试顺序将大大减少集成测试和回归测试的工作量。本文介绍的基于测试级的集成测试顺序分配策略和测试工具 TLOGOS, 已被用于电力监控组态系统的测试中。结果表明: 按照该工具产生的测试顺序对系统进行测试, 比随机顺序测试节省约 40% 的工作量。

参考文献 (References)

[1] Jorgensen P C, Erickson C. Object-oriented integration testing[J]. CACM, 1994, 37(9): 30-38.

[2] Kung D C. Class firewall, test order, and regression testing of object-oriented programs[J]. Journal of Object-oriented Programming, 1995, 8(2): 51-65.

[3] Murphy G C, Townsend P, Wong P S. Experience with cluster and class testing[J]. CACM, 1994, 37(9): 39-47.

[4] Kung D C. Developing an object-oriented software testing and maintenance environment[J]. Communications of the ACM, 1995, 38(10): 75-87.

[5] Fewster M. 软件测试自动化技术与实例详解[M]. 苏智勇译. 北京: 电子工业出版社, 2001. 75-78.

(责任编校: 孙陆青)